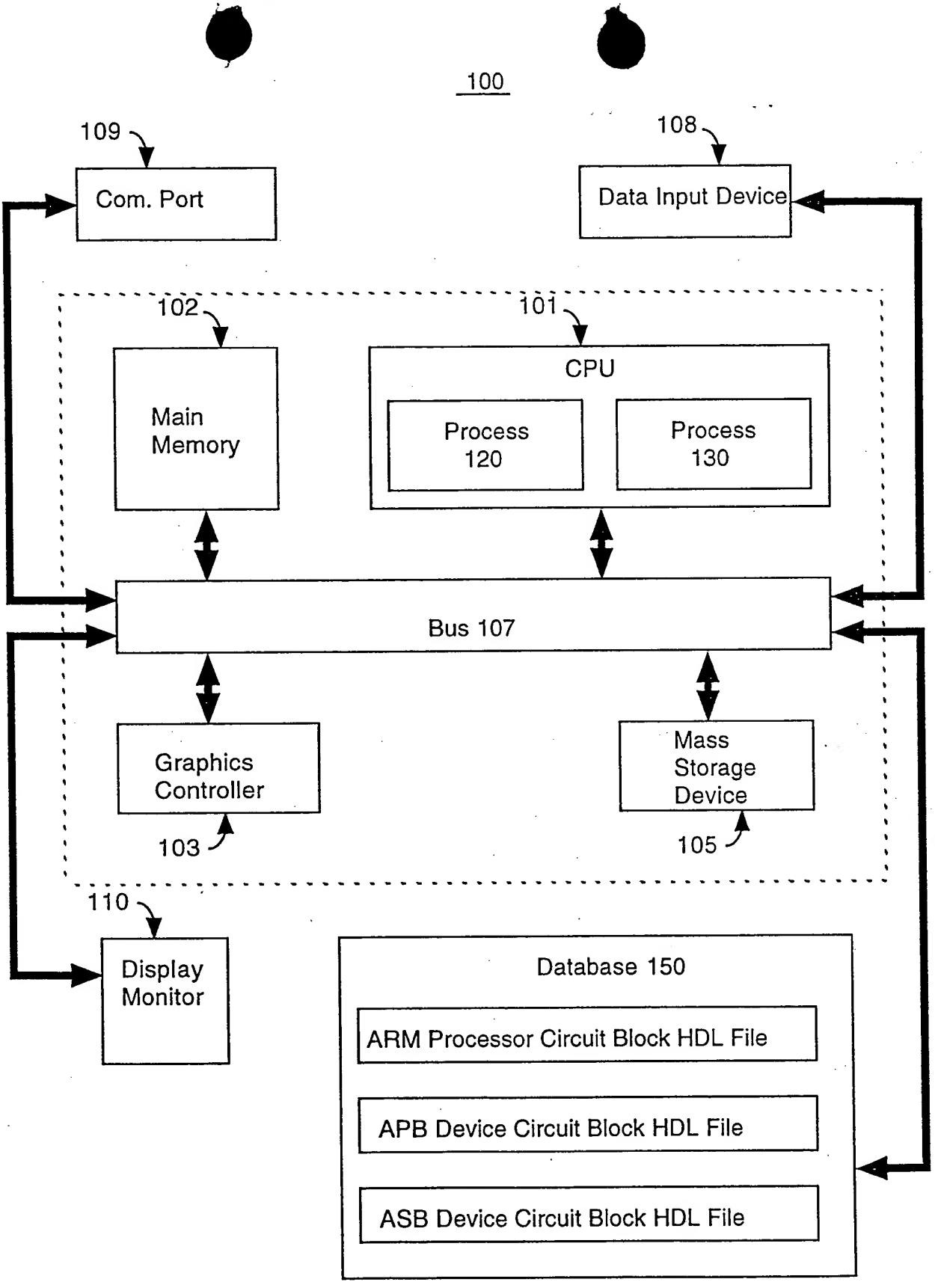

100
109
Com. Port
108
Data Input Device
102
Main Memory
101
CPU
Process 120
Process 130
Bus 107
Graphics Controller
103
Mass Storage Device
105
110
Display Monitor
Database 150
ARM Processor Circuit Block HDL File
APB Device Circuit Block HDL File
ASB Device Circuit Block HDL File

FIG. 1

200

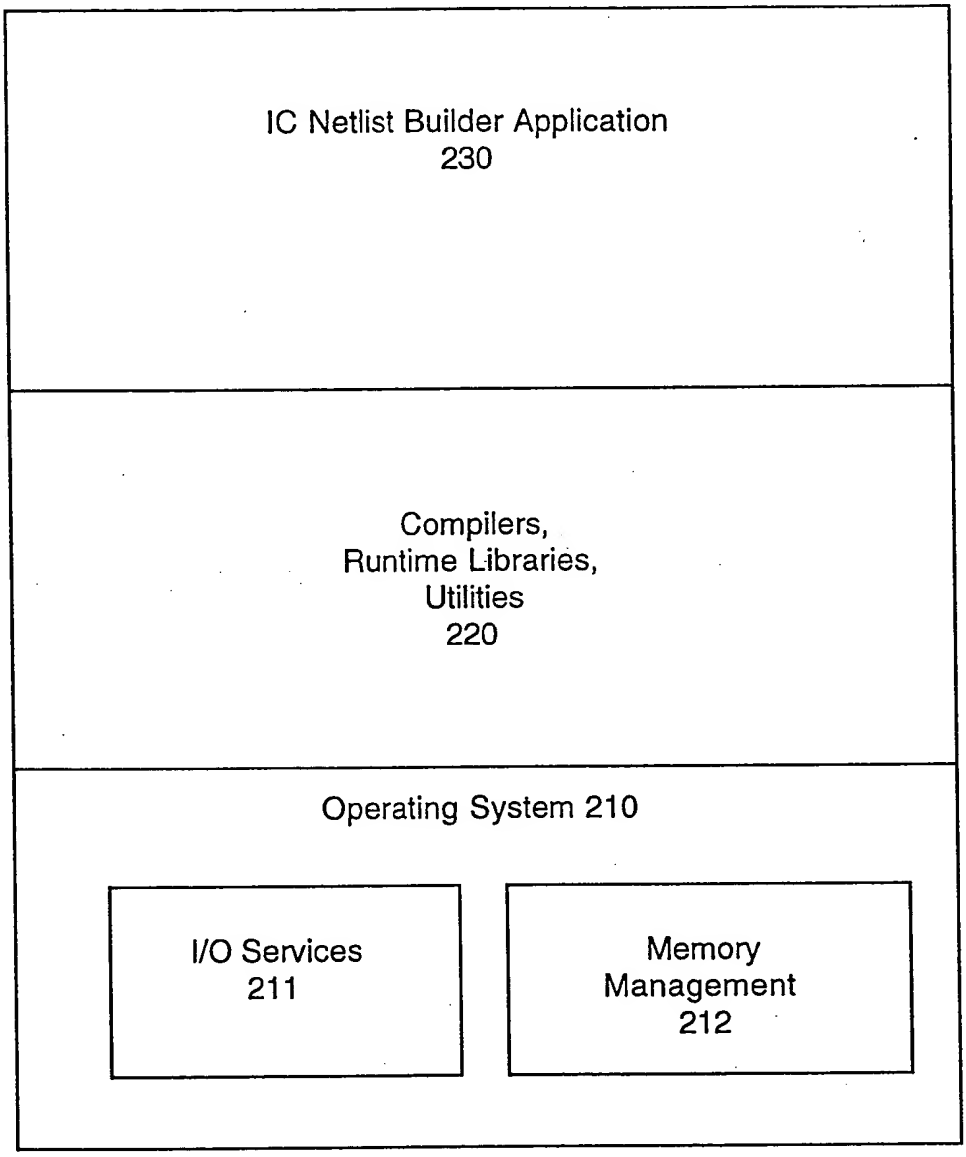

IC Netlist Builder Application
230
Compilers,
Runtime Libraries,
Utilities
220
Operating System 210
I/O Services
211
Memory
Management
212

FIG 2

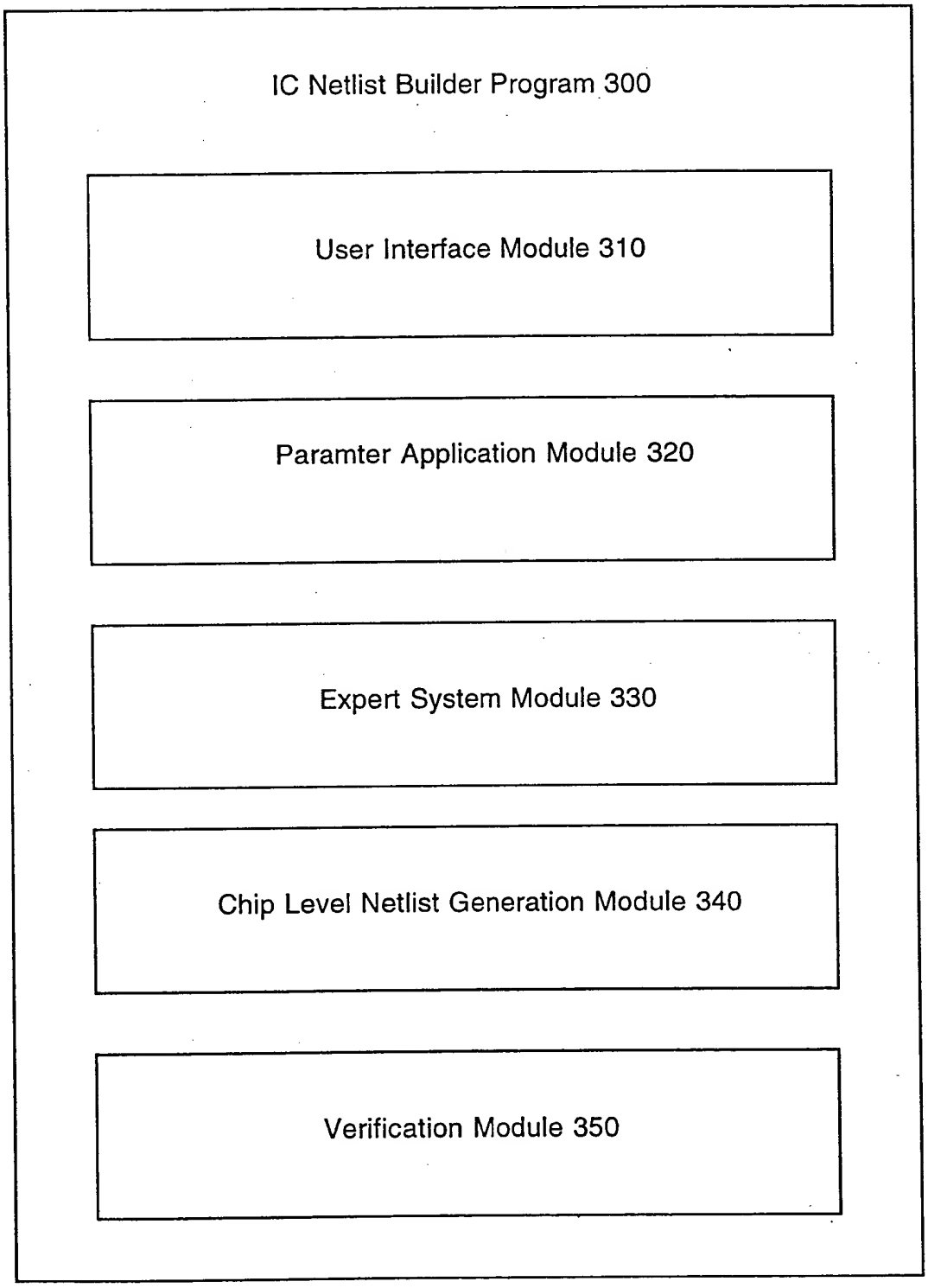

IC Netlist Builder Program 300
User Interface Module 310
Paramter Application Module 320
Expert System Module 330
Chip Level Netlist Generation Module 340
Verification Module 350

FIG 3

```
proc do_timers{}
{
        if { $ { : : VPB0.TIMERS_numelements} > 0 } {
        set DSGN_PARAM "timer -tsize 32 -psize 32 -psize 32 -pcons '0' -pdmax 1 -nmr
NMR -extm Y_N_MASK -nc r NCR -t 2 -areset 2 -0 TIMERNAME"
        set i 0
        while {${ : : VPBO.TIMERS_numelements > $i} {
                set DSGN_PARAM_COPY $DSGN_PARAM
                #-- if we have only one timer we make sure
                #-- that we do not use index elements
                if {${ : : VPBO.TIMERS_numelements} = = 1} {
                set devicename "timer"
                set  tmpstr ""
        } else {
            set devicename "timer$i"
            set tmpstr "$i"
        }
```

FIG 4A

```
        regsub -all "NMR" $DSGN_PARAM_COPY [set
::VPB0.TIMERS$i.NUM_MATCH] DSGN_PARAM_COPY
        regsub -all "Y_N_MASK" $DSGN_PARAM_COPY [set
::VPB0.TIMERS$i.MATCH_INT] DSGN_PARAM_CO

PY

        regsub -all "NMR" $DSGN_PARAM_COPY [set ::VPB0.TIMERS$i.NCR]
DSGN_PARAM_COPY
        regsub -all "TIMERNAME" $DSGN_PARAM_COPY $ (devicename}
DSGN_PARAM_COPY
```

FIG 4B

```
                lappend : : library_list "library $ {devicename}_lib; \nuse
${devicename}_lib.$(devicename)_pkg.all; \n"
                lappend: : vpb_dev_name $ {devicename}
                lappend: :  gate_count  [list - n "Timer $i " -g 6135 -sg " " -t Sy ]
```

FIG 4C

```
        lappened : : sgnl_name "VPB Timer $devicename"
        lappened : : sgnl_name "tmr$ {tmpstr}_pause | 1| 1 "
        lappened : : clk_dst_name "ct_tmr${devicename}_pclk | ck_nbclk"
        set iinfo ""
        lappened iinfo "u_tmr$tmpstr | timer$tmpstr | VPB Timer $tmpstr"
        lappened iinfo "pclk | ct_tmr$ {tmpstr}_pclk | VPB Bus Clock"
        lappened iinfo "pstb | pstb | VPB Peripheral Strobe "
        lappened iinfo "psel | psel_tmr${tmpstr} | VPB Peripheral Select"
        lappened iinfo "pwrite | pwrite | VPB Peripheral Write"
        lappened iinfo "pd | pd | VPB Data Bus (31:0)"
        if { [set : : VPB0.TIMERS$i.NCR] > 0 } {
            lappend : : assign_list " Tie off capture Inputs "
        lappend : : assign_list "tmr$ {tmr$(tmpstr}_capture | (expr (($: : LANGUAGE) = =
(LANG_VHDL ) ) ?{ (others =>logic_01) : {logic_0}] "
        lappend iinfo "capture | tmr${(tmpstr)_capture | Timer Capture Signals"
        lappened : : sgnl_name "tmr$ {tmpstr}_capture | set : : VPB0.TIMERS$i.NCR] |"
}
if { [set : : VPB0.TIMERS$i.NUM_MATCH] > 0} {
        lappened iinfo "nint | tmr${tmpstr}_int | Timer Interrupt, Active Low"
        lappened : : intr_src "${devicename}_nint"
}
it { [set : : VPB0.TIMERS$i.NCR] > 0 / / [set : : VPB0.TIMERS$i.NUM_MATCH] > 0} {
        lappened iinfo "pnres / ct_tmr$(tmpstr)_pures | VPB Asynchronous Timer Reset"
        lappened iinfo "pa | pa(5 downto 0) | VPB Address Bus (5:0)"
        lappened : : rst_dst_name "rs_tmr$(tmpstr)_pnres | cg_nbclk"
} else {
        lappened iinfo "pa | pa(4 downto 0) | VPB Address Bus (5:0)"
}
lappend iinfo "pause | tmr$(tmpstr)_pause | Timer Pause"
lappend : : assign_list "Tie off pause Inputs "
lappend : : assign_list " tmr$(tmpstr)_pause | logic_0"
#- - Based on the number of match outputs, we create interrupt sources
if {[set : : VPB0.TIMERS$i.MATCH_INT] = = 1}  {
    set j 0
    while { [set : : VPB0.TIMERS$i.NUM_MATCH] > $j} {
        lappend : : intr_src "${devicename}_m$j"
        lappend iinfo "m$j | tmr${tmpstr}_m$j | Timer $tmpstr External Match $j Output

        incr j
    }
}
lappend iinfo "scantestmode | scantestmode | Scan Test Mode"
lappend  : : inst_list $iinfo
```

FIG 5

```
proc create_chipcore {}  {
      set ccfid [open "$ : : WORK_DIR/ S : : COMPONENTNAME/chip/top/chipcore.v"
"a"]
#  Module and port type declaration has already been written by padring.tcl
         foreach elem $ : : sgnl_name  {
            regsub  { [ / t] +$} $elem { } elem
            set selem [split $elem " / " ]
            switch [llength $selem] {
                1  { puts $ccfid "$ { : :comment} [string trim [lindex $selem 0] ]" }
                 2  { set swidth [lindex $selem 1] ; puts $ccfid [format "  wire %7s %s" [expr
($swi
dth= = 1) ? { }: {\[[ expr $swidth-1] \:0 \]}] "string trim [lindex $selem 0] ] ;" ] }
                 3  { set swidth {lindex $selem 1]; puts $ccfid [format "  wire %7s %-40s ${: :
comme
nt) %s" [expr ($swidth= = 1) ? { }: {\[ [ expr $swidth- 1 ] \: 0\]}] "string trim [lindex $selem
0]]; "  [strin
g trim [lindex $selem 2 ] ] ] }
      }
   }
```

FIG 6A

```
     puts $ccfid "${: : comment} Reset and Clock signals"
foreach elem [ concat $ : : rst_dst_name $: : clk_dst_name] {

         set selem [split $elem "/ " ]
         puts $ccfid [ccfid [format "   wire          %s; " [string trim [lindex $selem 0] ] ]
}
puts $ccfid "\n[string repeat $: : comment 30]"
puts $ccfid "$ : : comment Assign statements"
puts $ccfid "\n[string repeat $: : comment 30] \n"
```

FIG 6B

```
#  Based on the assign_list, generate the HDL assign statements as required.
       foreach elem $ : : assign_list  {
           regsub  { [  /\t] + $elem { } elem
           set aelem [split $elem "/ "]
           if { [llength $aelem]==1}  {
                 puts $ccfid "${ : : comment} [lindex $aelem 0]"
           } else  {
               if  {{$: : LANGUAGE}= = "LANG_VHDL"}  {
                    puts $ccfid [format "  %15s <= %s; " [string trim [lindex $aelem 0 ] ]
[string trim [lindex $aelem 1 ]]]
                          } else {
                              puts $ccfid [format " assign %15s = %s; " [string trim [lindex
$aelem 0 ] ] [string trim [lindex $aelem 1]]]
                          }
                 }
         }
```

FIG 6C

```
proc create_chipcore {}  {
      set ccfid [open "$ : : WORK_DIR/ S : : COMPONENTNAME/chip/top/chipcore.v"
"a"]

# Module and port type declaration has already been written by padring.tcl

# Based on the sgnl_name list, which contains all the internal signals
# that must be generated for proper connectively, create the actual
# VHDL or Verilog code that will perform that task.
        foreach elem $ : : sgnl_name  {




    foreach iinfo  $: : inst_list  {
        set first 1
        set istr " "
        foreach pmap $iinfo  {
            regsub -all  { [ \t] *\ / [ \t]*} $pmap {/ } pmap
            set pelem [split $pmap  {/ }]
            if  { $first  }  {
                set first 0
                    append istr "\n${ : : comment) (string repeat $ {: : comment} 37] \n"
                  append istr  "${ : : comment}  [lindex $pelem 2]\n"
                   append istr "${ : : comment) (string repeat S {: : comment} 37] \n"
              append istr "    [lindex $pelem 1 ]  [lindex $pelem 0]   (\n"
         } else {
             set formal [lindex $pelem 0]
             set actual [lindex $pelem 1]
             if {{$ : : LANGUAGE} = = "LANG_VERILOG"} {
                 regsub -all  {\ ( }  $actual {[ } actual
                 regsub -all  {\ ( }  $actual {] } actual
                     regsub {open} $actual { } actual
                 regsub - all "downto " $actual ": " actual
                 regsub {\ (} $formal {[} formal
                 regsub {\ )} $formal {]} formal
                 regsub " downto " $formal ":" formal
            }
            append istr "        .$formal ($actual) , \n"
        }

        regsub {, \n$}  $istr "  ) ; " istr
        puts $ccfid $istr
        }
        puts $ccfid "\n\nendmodule \n"
```

FIG 7

800

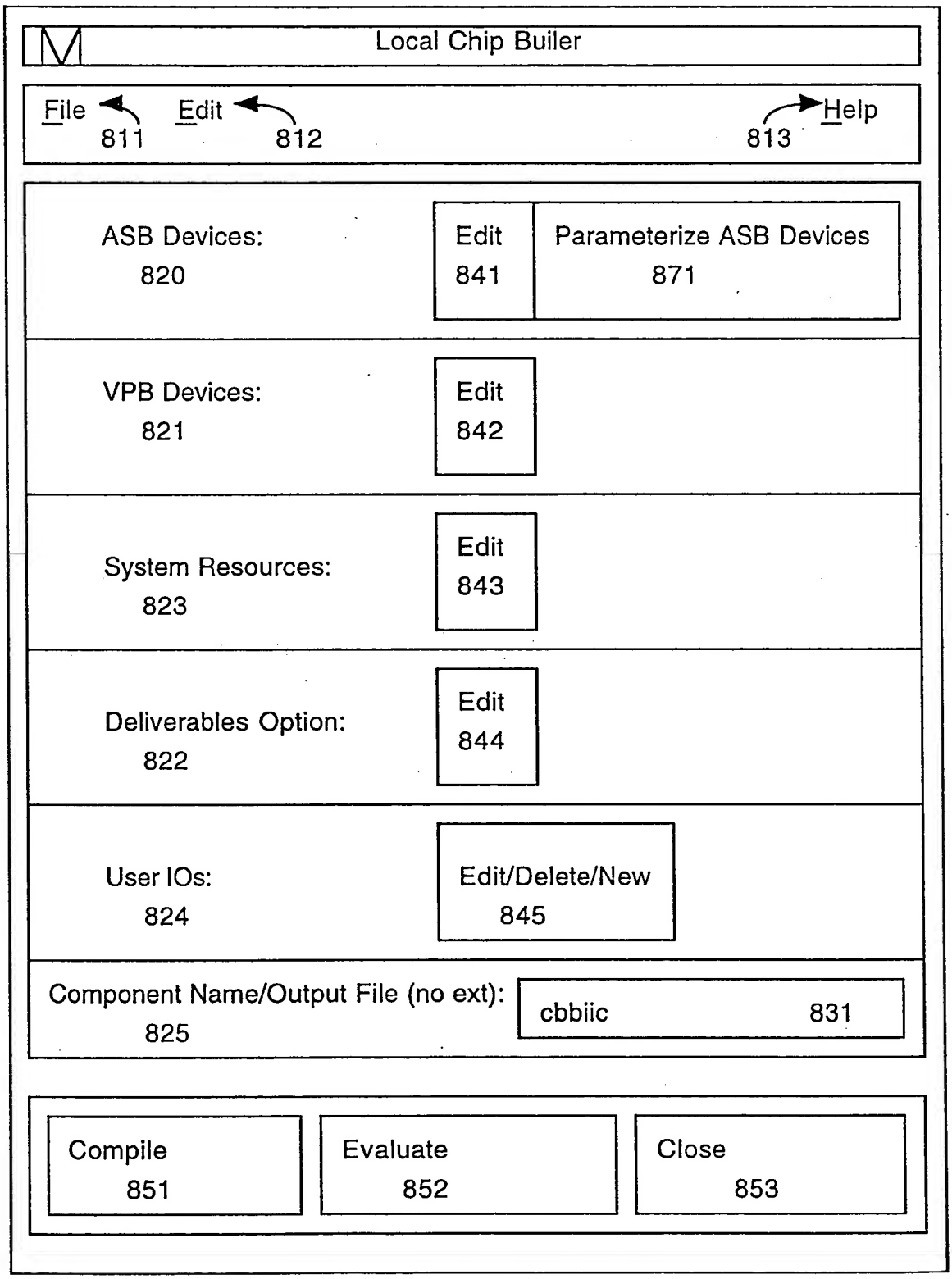

Local Chip Builer
File
811
Edit
812
Help
813
ASB Devices:
820
Edit
841
Parameterize ASB Devices
871
VPB Devices:
821
Edit
842
System Resources:
823
Edit
843
Deliverables Option:
822
Edit
844
User IOs:
824
Edit/Delete/New
845
Component Name/Output File (no ext):
825
cbbiic
831
Compile
851
Evaluate
852
Close
853

Fig 8

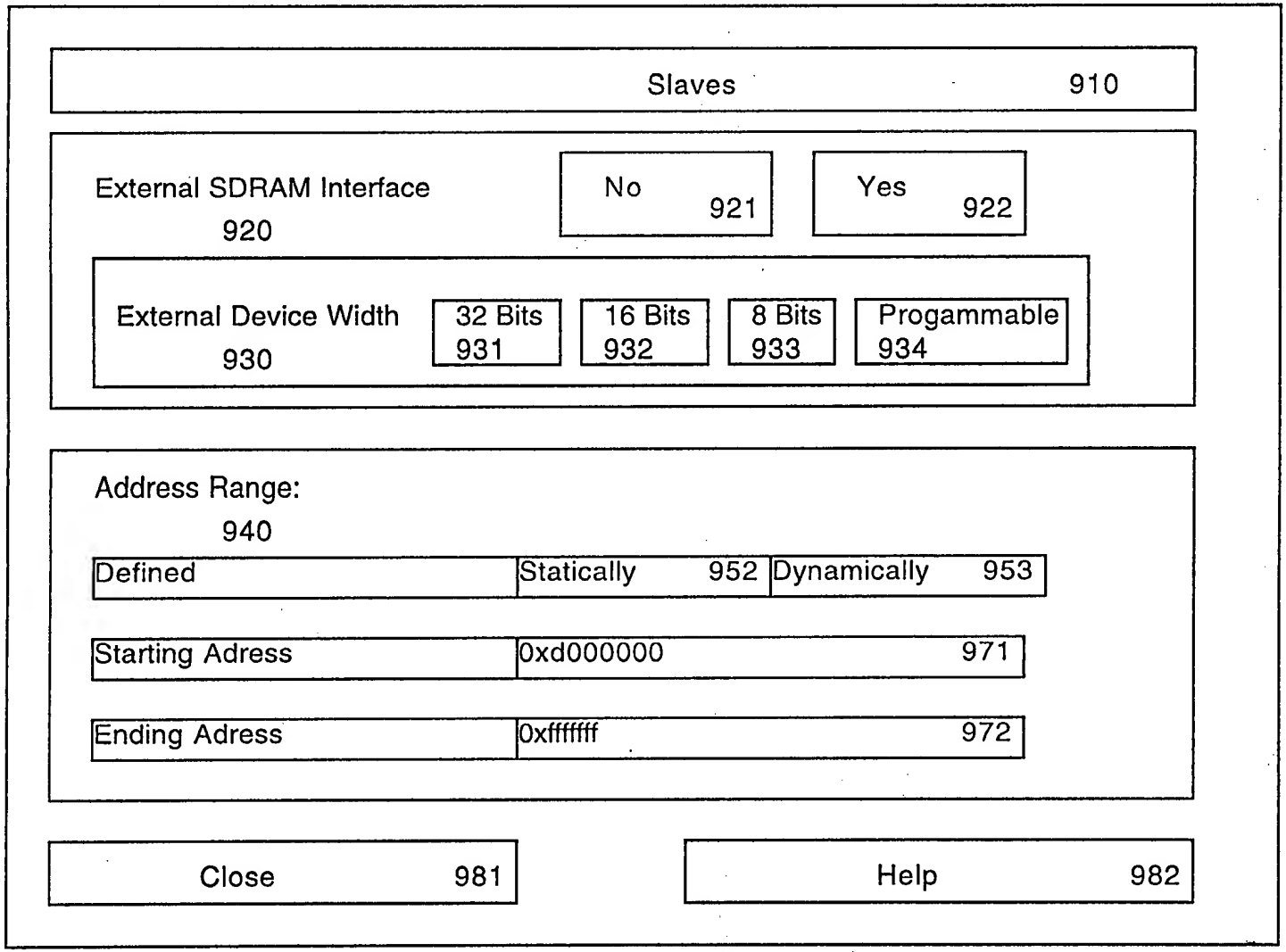

Slaves 910
External SDRAM Interface 920
No 921
Yes 922
External Device Width 930
32 Bits 931
16 Bits 932
8 Bits 933
Progammable 934
Address Range: 940
Defined
Statically 952
Dynamically 953
Starting Adress
0xd000000 971
Ending Adress
0xffffffff 972
Close 981
Help 982

Fig 9

1000

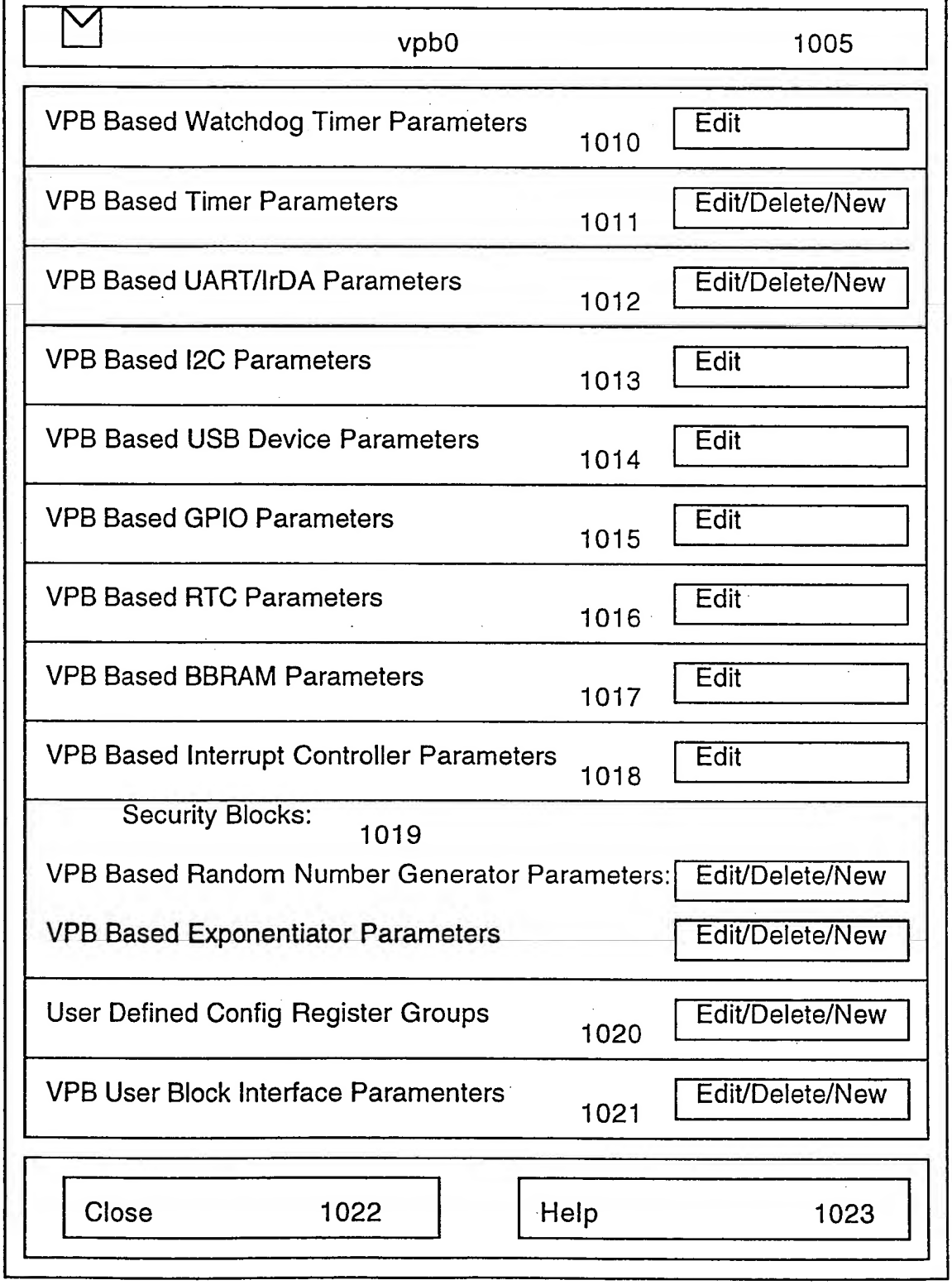

vpb0
1005
VPB Based Watchdog Timer Parameters
1010
Edit
VPB Based Timer Parameters
1011
Edit/Delete/New
VPB Based UART/IrDA Parameters
1012
Edit/Delete/New
VPB Based I2C Parameters
1013
Edit
VPB Based USB Device Parameters
1014
Edit
VPB Based GPIO Parameters
1015
Edit
VPB Based RTC Parameters
1016
Edit
VPB Based BBRAM Parameters
1017
Edit
VPB Based Interrupt Controller Parameters
1018
Edit
Security Blocks:
1019
VPB Based Random Number Generator Parameters:
Edit/Delete/New
VPB Based Exponentiator Parameters
Edit/Delete/New
User Defined Config Register Groups
1020
Edit/Delete/New
VPB User Block Interface Paramenters
1021
Edit/Delete/New
Close
1022
Help
1023

Fig 10

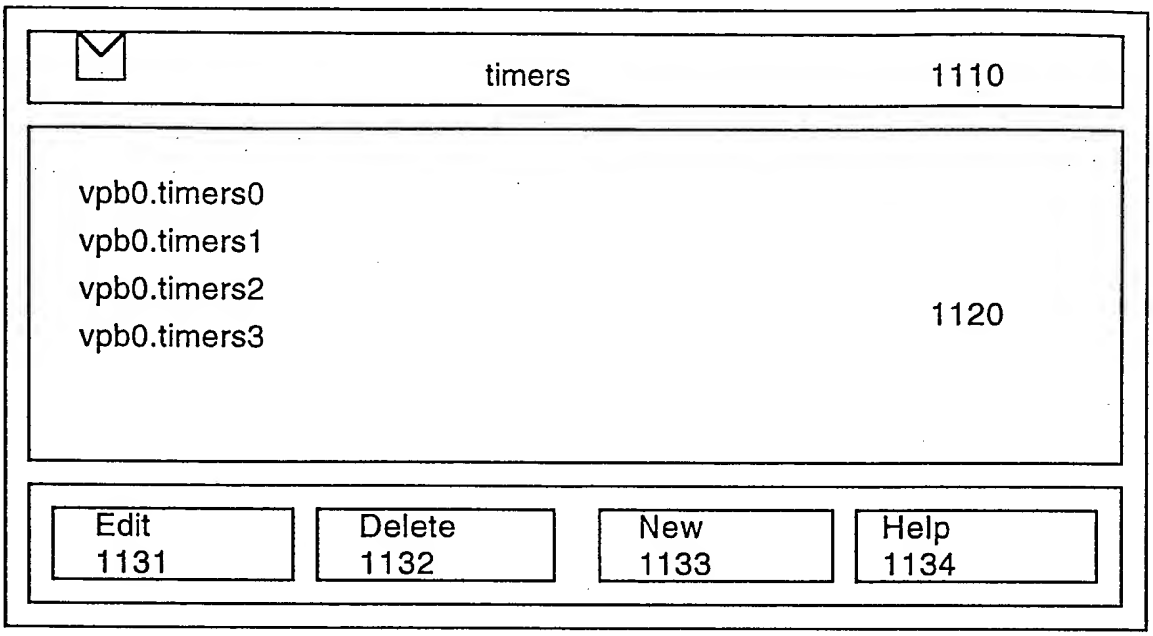

timers
1110
vpb0.timers0
vpb0.timers1
vpb0.timers2
vpb0.timers3
1120
Edit
1131
Delete
1132
New
1133
Help
1134

FIG 11

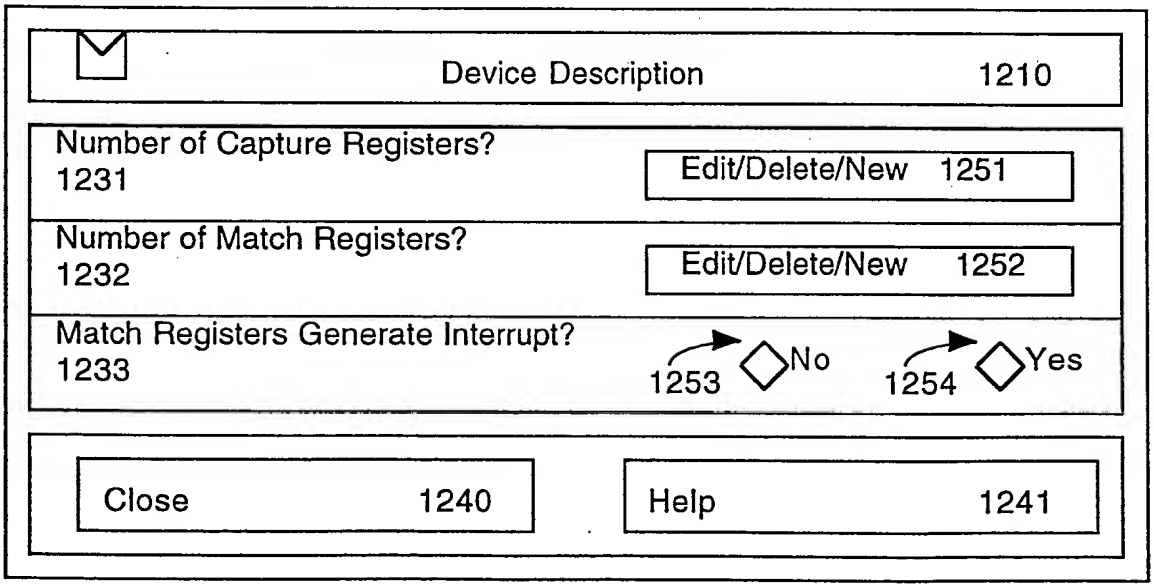

Device Description
1210
Number of Capture Registers?
1231
Edit/Delete/New 1251
Number of Match Registers?
1232
Edit/Delete/New 1252
Match Registers Generate Interrupt?
1233
1253
No
1254
Yes
Close
1240
Help
1241

Fig 12

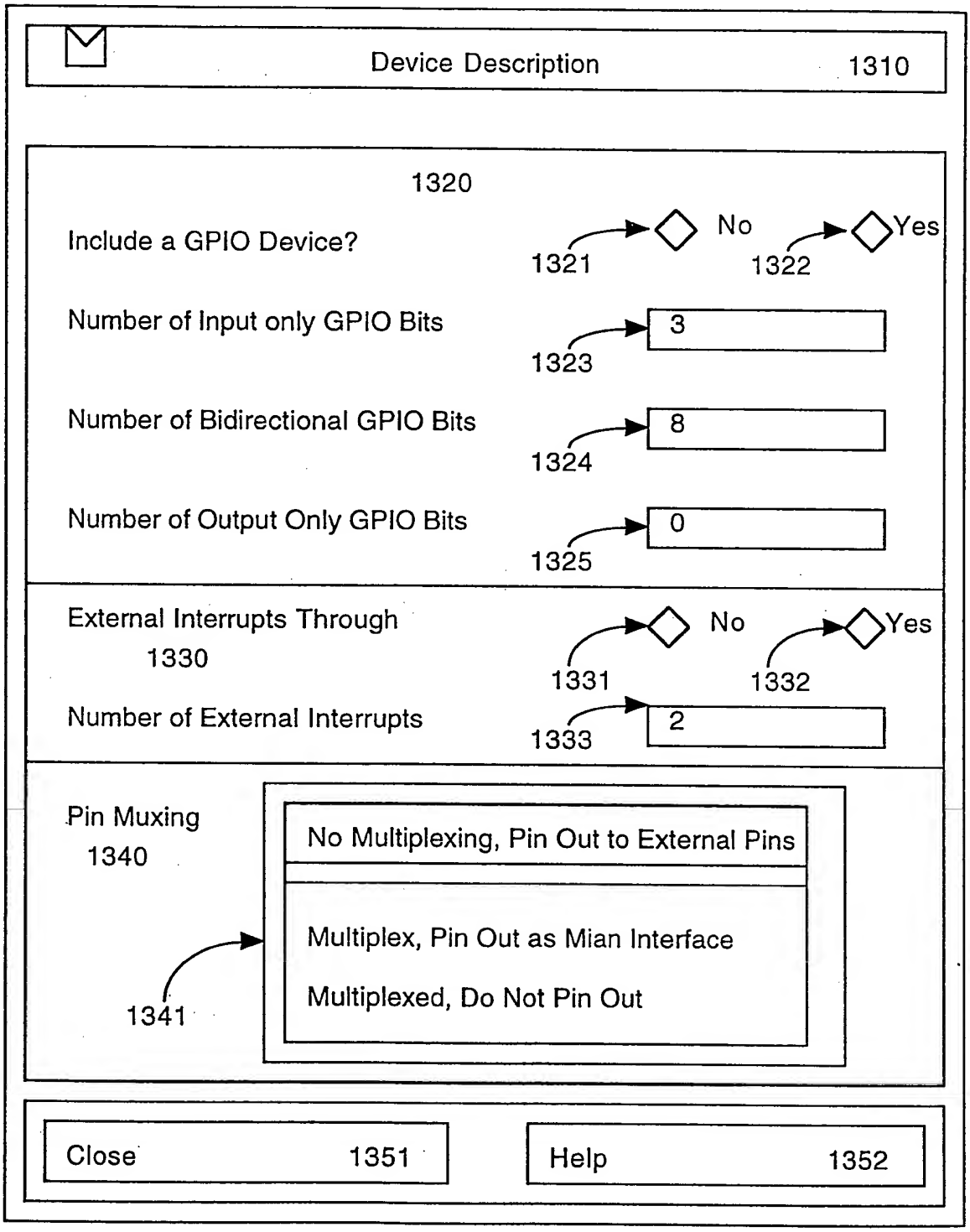
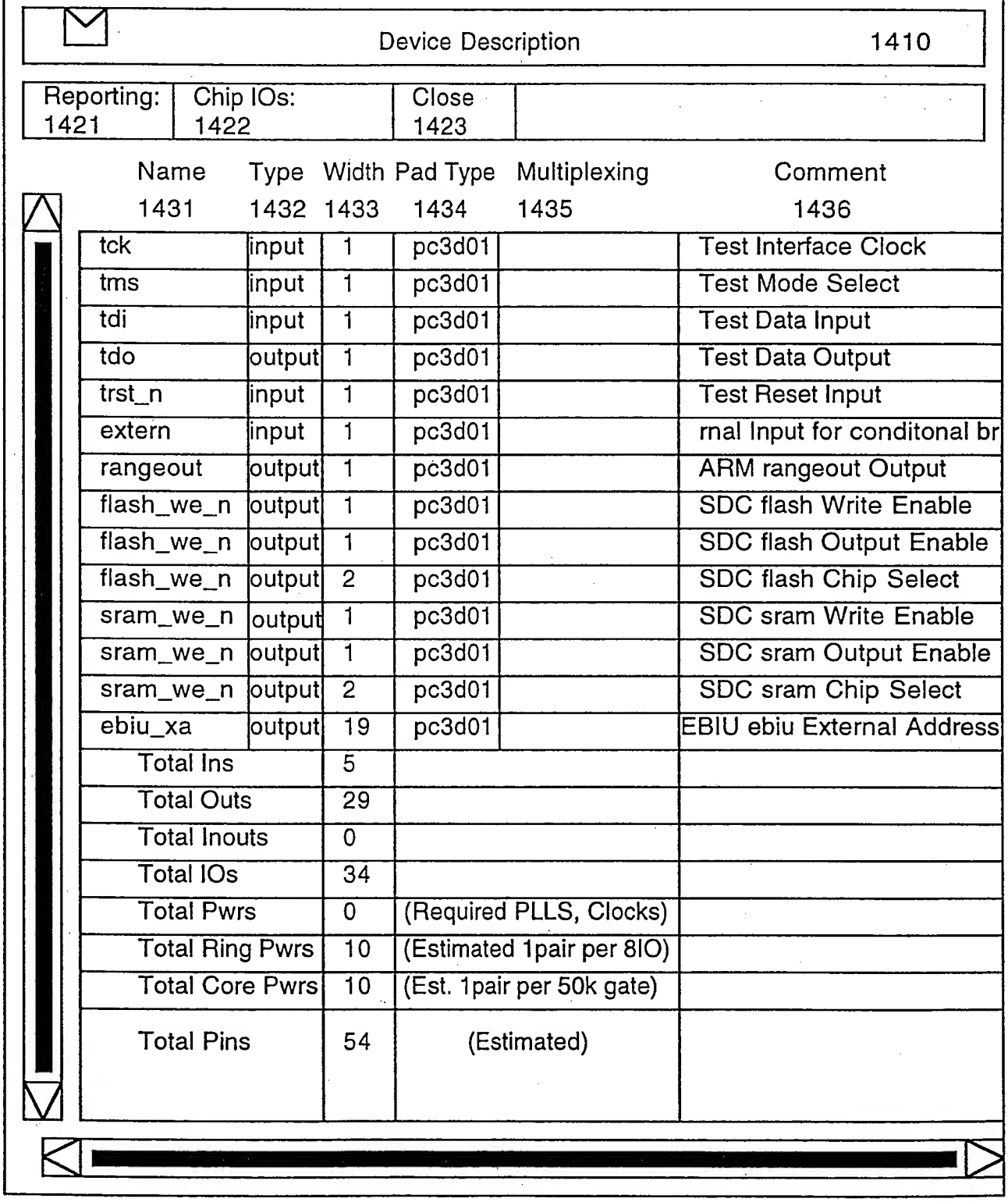

Device Description
1310
1320
Include a GPIO Device?
No
Yes
1321
1322
Number of Input only GPIO Bits
3
1323
Number of Bidirectional GPIO Bits
8
1324
Number of Output Only GPIO Bits
0
1325
External Interrupts Through
1330
No
Yes
1331
1332
Number of External Interrupts
2
1333
Pin Muxing
1340
No Multiplexing, Pin Out to External Pins
Multiplex, Pin Out as Mian Interface
Multiplexed, Do Not Pin Out
1341
Close
1351
Help
1352

Fig 13

1400



Name 1431	Type 1432	Width 1433	Pad Type 1434	Multiplexing 1435	Comment 1436
tck	input	1	pc3d01		Test Interface Clock
tms	input	1	pc3d01		Test Mode Select
tdi	input	1	pc3d01		Test Data Input
tdo	output	1	pc3d01		Test Data Output
trst_n	input	1	pc3d01		Test Reset Input
extern	input	1	pc3d01		rnal Input for conditonal br
rangeout	output	1	pc3d01		ARM rangeout Output
flash_we_n	output	1	pc3d01		SDC flash Write Enable
flash_we_n	output	1	pc3d01		SDC flash Output Enable
flash_we_n	output	2	pc3d01		SDC flash Chip Select
sram_we_n	output	1	pc3d01		SDC sram Write Enable
sram_we_n	output	1	pc3d01		SDC sram Output Enable
sram_we_n	output	2	pc3d01		SDC sram Chip Select
ebiu_xa	output	19	pc3d01		EBIU ebiu External Address
Total Ins		5			
Total Outs		29			
Total Inouts		0			
Total IOs		34			
Total Pwrs		0	(Required PLLS, Clocks)		
Total Ring Pwrs		10	(Estimated 1pair per 8IO)		
Total Core Pwrs		10	(Est. 1pair per 50k gate)		
Total Pins		54	(Estimated)		

Fig 14

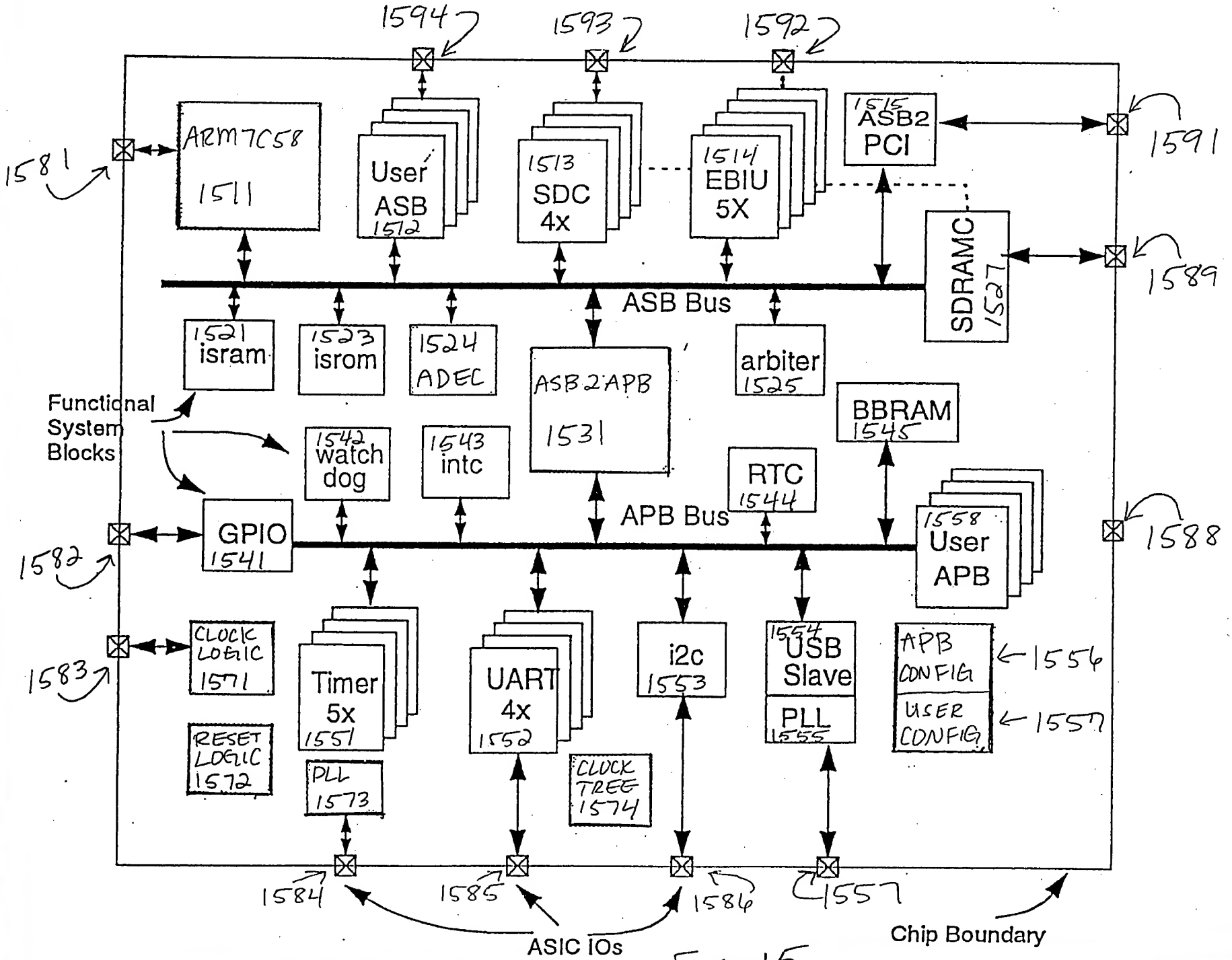

1594
1593
1592
1581
ARM7C58
1511
User ASB
1512
1513
SDC 4x
1514
EBIU 5X
1515
ASB2 PCI
1591
1589
SDRAMC
1527
ASB Bus
1521
isram
1523
isrom
1524
ADEC
ASB2APB
1531
arbiter
1525
Functional System Blocks
1542
watch dog
1543
intc
BBRAM
1545
RTC
1544
1582
GPIO
1541
APB Bus
1558
User APB
1588
1583
CLOCK LOGIC
1571
Timer 5x
1551
UART 4x
1552
i2c
1553
1554
USB Slave
PLL
1555
APB CONFIG
1556
USER CONFIG
1557
RESET LOGIC
1572
PLL
1573
CLOCK TREE
1574
1584
1585
1586
1557
ASIC IOs
Chip Boundary

FIG 15

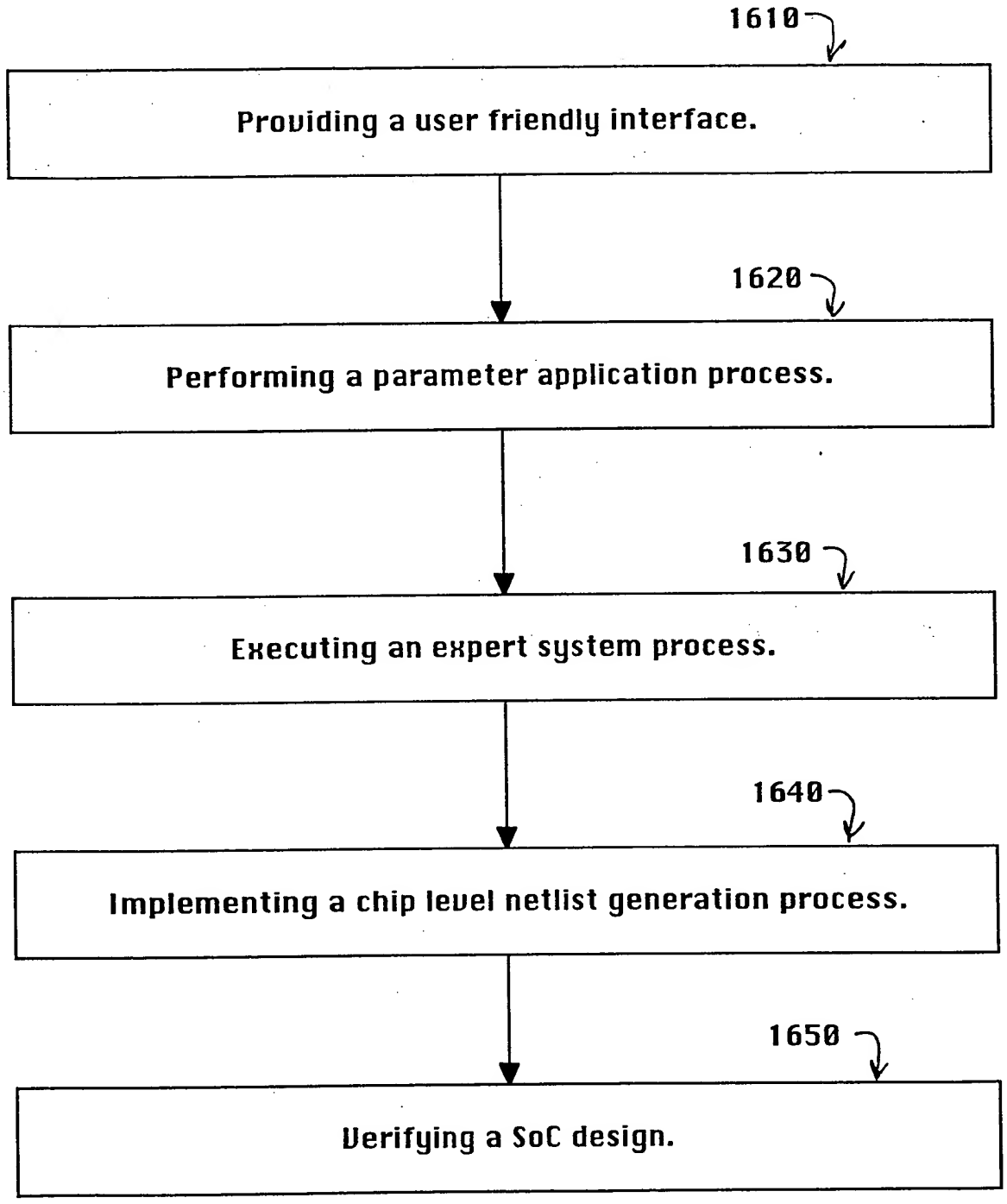

1600
1610
Providing a user friendly interface.
1620
Performing a parameter application process.
1630
Executing an expert system process.
1640
Implementing a chip level netlist generation process.
1650
Verifying a SoC design.

FIG 16

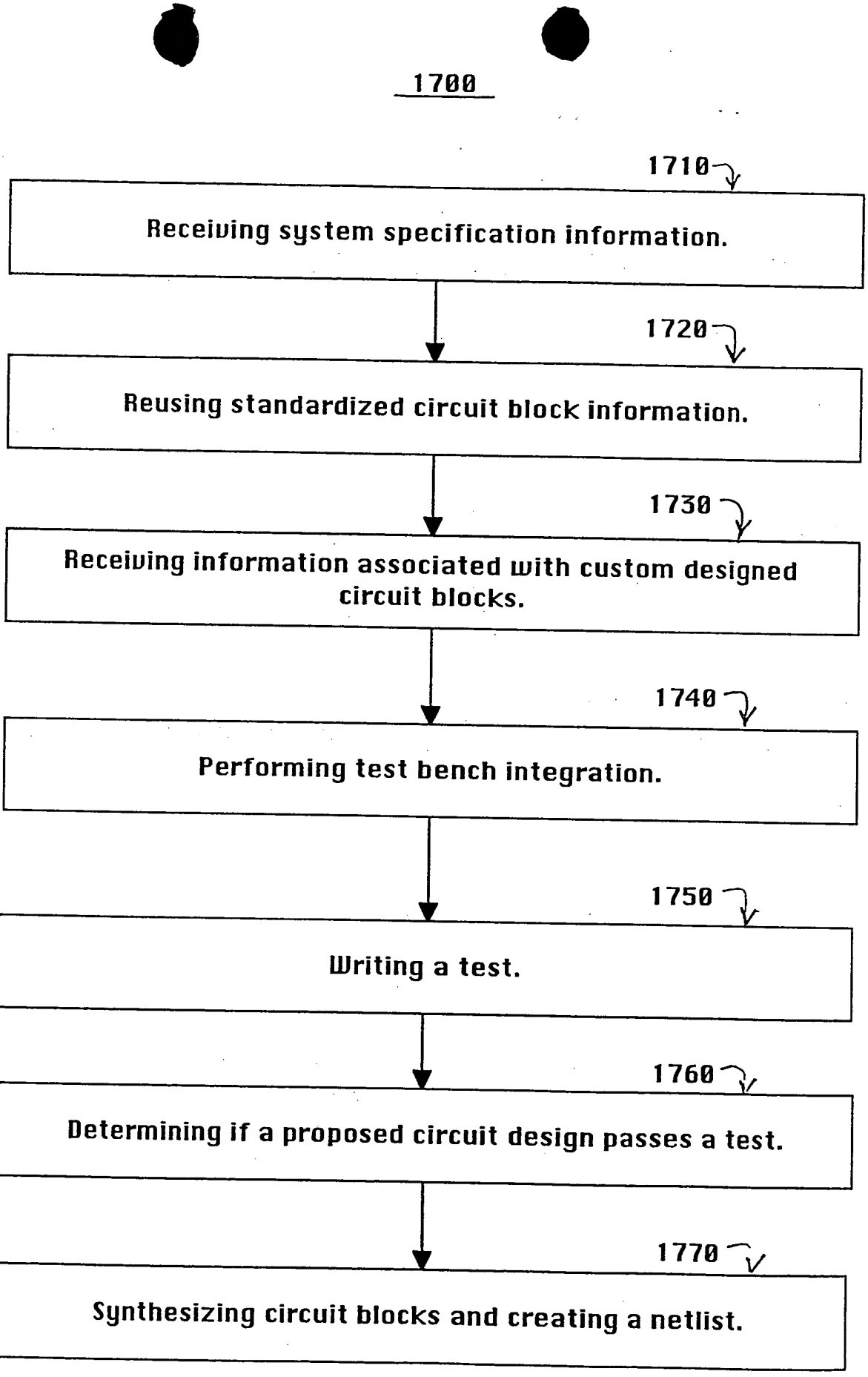

1700
1710
Receiving system specification information.
1720
Reusing standardized circuit block information.
1730
Receiving information associated with custom designed circuit blocks.
1740
Performing test bench integration.
1750
Writing a test.
1760
Determining if a proposed circuit design passes a test.
1770
Synthesizing circuit blocks and creating a netlist.

FIG 16